Comprehensibility Improvement of Tabular Knowledge Bases

Atsushi Sugiura† **Maximilian Riesenhuber**‡ **Yoshiyuki Koseki**†

†C&C Systems Res. Lab. NEC Corp.
4-1-1 Miyazaki Miyamae-ku
Kawasaki 216 JAPAN
sugiura@btl.cl.nec.co.jp

‡Inst. of Theoretical Physics
Johann Wolfgang Goethe-Univ.
Robert-Mayer-Strasse 8-10
Frankfurt am Main 60054 Fed. Rep. of GERMANY

Abstract

This paper discusses the important issue of knowledge base comprehensibility and describes a technique for comprehensibility improvement. Comprehensibility is often measured by simplicity of concept description. Even in the simplest form, however, there will be a number of different DNF (Disjunctive Normal Form) descriptions possible to represent the same concept, and each of these will have a different degree of comprehensibility. In other words, simplification does not necessarily guarantee improved comprehensibility. In this paper, the authors introduce three new comprehensibility criteria, *similarity*, *continuity*, and *conformity*, for use with tabular knowledge bases. In addition, they propose an algorithm to convert a decision table with poor comprehensibility to one with high comprehensibility, while preserving logical equivalency. In experiments, the algorithm generated either the same or similar tables to those generated by humans.

Introduction

Two major requirements for knowledge base are that it contain only correct knowledge and that it be comprehensible. Several techniques have been reported regarding the verification of correctness, including the completeness and consistency checking [Cragun 1987, Nguyen et al. 1985]. However, little work has been reported concerning the maintenance or improvement of comprehensibility.

Comprehensibility is critical, because it strongly affects efficiency of construction and maintenance of knowledge bases. However, the actual work of modifying knowledge descriptions so as to improve the comprehensibility can prove to be a serious burden for the knowledge engineers who must manage knowledge bases. Purpose of this research is to automate such tasks.

In past work [Michalski, Carbonell, & Michell 1983, Coulon & Kayser 1978], the one and only method to improve the comprehensibility of a knowledge base was to simplify the concept descriptions in it. Even in the simplest form, however, there will be a number of different DNF descriptions possible to represent the same concept, and each of these will have a different degree of comprehensibility. In other words, simplification does not necessarily guarantee improved comprehensibility.

Let us compare the following two logic functions of attribute-value:

$[\textit{Sex Male}] \vee [\textit{Sex Female}] \wedge [\textit{Pregnant? No}]$
$\rightarrow \textit{Can\_Drink\_Alcohol}$

$[\textit{Pregnant? No}] \vee [\textit{Pregnant? Yes}] \wedge [\textit{Sex Male}]$
$\rightarrow \textit{Can\_Drink\_Alcohol}$

While these are logically equivalent and have the same simplicity, their *concept function forms*, that is formalized by the combination of attribute values, conjunctions ($\wedge$) and disjunctions ($\vee$), are different. The second description is incomprehensible, because it describes a case that never happens: $[\textit{Pregnant? Yes}] \wedge [\textit{Sex Male}]$.

In this paper, the authors propose three additional comprehensibility criteria for use in concept function forms on decision tables: similarity among concept functions, continuity in attributes which have ordinal values, and conformity between concept functions and real cases. The first criterion is developed on the basis of an analysis of decision table characteristics. The others are developed on the basis of consideration of meaning embodied in expressions of knowledge.

In addition, the authors describe an algorithm to convert a decision table with poor comprehensibility to one with high comprehensibility, while preserving logical equivalency. This conversion is accomplished by using MINI-like logic minimization techniques [Hong, Cain, & Ostapko 1974], and it involves as well the use of a number of different heuristics.

Decision Table

With regard to knowledge acquisition, completeness checking, and concept comparison, decision tables offer advantages over other methods of knowledge representation (e.g. production rules and decision trees, etc.)

Table 1: Comprehensible expression for bond selection.

	Material			Usage		Bonding-area	
	Paper	Leather	Plastic	Normal	Industrial	Large	Small
Bond-A	○	×	×	○	○	○	○
Bond-B	○	×	×	○	○	○	○
Bond-B	×	○	×	○	×	○	○
Bond-B	×	○	×	×	○	○	×
Bond-C	×	×	○	○	×	○	○

Table 2: Incomprehensible expression for bond selection.

	Material			Usage		Bonding-area	
	Paper	Leather	Plastic	Normal	Industrial	Large	Small
Bond-A	○	×	×	○	○	○	○
Bond-B	○	×	×	×	○	×	○
Bond-B	○	○	×	○	○	○	×
Bond-B	○	○	×	○	×	×	○
Bond-C	×	×	○	○	×	○	○

[Cragun 1987, Koseki, Nakakuki, & Tanaka 1991].

A decision table represents a set of concept functions, expressed in DNF. This construct enables the handling of disjunctive concepts which have multiple-value attributes. Each concept function consists of a number of disjuncts, called *cubes*. Each cube consists of a set of values for an attribute. The union of the vertices in logic space, covered by concept function, is called a *cover* for the concept.

An example knowledge base is shown in Table 1. Here, each row forms a cube, and a set of cubes for the same concept name forms a a concept function. For example, the concept function for *Bond-B* consists of three cubes, and each cube consists of three attributes. In a cube, the values specified by the circle (○) in an attribute are ORed, and all attributes are ANDed to form the cube. Don't-Care attributes are designated as an attribute with all ○s. A min-term is a cube which has only one ○ in every attribute.

The decision table facilitates the comparison of different concepts, due to the following reason. Concept comparison means comparing the values of all the attributes which define concepts. In this context, the decision tables, in which description of the same attribute is represented in the same columns and concepts are represented by all attributes, can facilitate concept comparison. In other knowledge representations, for example in the production rules, the same attribute name appears in various positions of each rule, and concepts are represented only by attributes that are necessary to define concepts. This disturbs easy comparison of attribute values.

This advantage is critical for the knowledge base constructions, because, in the classification problems, it is essential to compare and to classify the concepts which have the same cover.

Comprehensibility Criteria

This section presents four criteria for knowledge base comprehensibility. One is simplicity of concept description, which is the conventional criterion. The other three are concerned with concept function forms. Three new criteria are a reflection of the great influence of the concept function forms on its comprehensibility.

The second criteria employs general rules to facilitate the comparison of different concepts, whereas the last two require some background knowledge, which is characteristics of attributes and their values.

Table Size

Preference criteria for human comprehensibility are commonly based on the knowledge description length (for example, [Michalski, Carbonell, & Michell 1983, Coulon & Kayser 1978]). Some inductive learning algorithms [Michalski, Carbonell, & Michell 1983, Quinlan 1986] reduce the knowledge base size to obtain simple knowledge expressions.

As the conventional criteria, the authors define table size as one of the comprehensibility factors. Since the number of attributes to define concepts is fixed in the decision tables, table size can be measured by the number of cubes.

Similarity among Concept Functions

When comparing different concepts, it is desirable to be able to easily ascertain common and different features in each concept. This requires high similarity among concept functions.

Table 1 and Table 2 are example knowledge bases for a bond selection problem. They have the same cover and the same table size, but their concept function forms (cube shapes) are different. In this example, Table 1 is better than Table 2, for the following reasons.

First, compare Bond-B with Bond-A. In the first cube (Bond-A) and the second cube (Bond-B) in Table 1, their forms are exactly the same. Using Table 1, the intersection of the two concepts in logic space can be determined by looking at just two cubes, whereas, using Table 2, this would require considering four cubes.

Next, compare Bond-B with Bond-C. In the third cube (Bond-B) and the fifth cube (Bond-C) in Table 1, descriptions of attributes *Usage* and *Bonding-area* are the same; only those of attribute *Material* are different. This makes it easy to see that these two concepts are discriminated by attribute *Material*. On the other hand, in Table 2, descriptions of most of the attributes are different, making the common and different features unclear.

Overall, comprehensibility of Table 1 arises from the high similarity among the concept function forms: the

Table 3: Comprehensible expression for Scholarship Approval.

	School-record		Student earn?		Parent-income (ten thousand dollars)			
	Good	Poor	Yes	No	-6	6-7	7-8	8-
Approved	○	×	×	○	○	○	○	○
Approved	○	×	○	×	○	○	○	×
Approved	×	○	×	○	○	○	×	×
Approved	×	○	○	×	○	×	×	×
Not_approved	○	×	○	×	×	×	×	○
Not_approved	×	○	×	○	×	×	○	○
Not_approved	×	○	○	×	×	○	○	○

Table 4: Incomprehensible expression for Scholarship Approval.

	School-record		Student earn?		Parent-income (ten thousand dollars)			
	Good	Poor	Yes	No	-6	6-7	7-8	8-
Approved	○	×	×	○	○	×	×	○
Approved	○	○	○	×	○	×	×	×
Approved	×	○	×	○	○	○	×	×
Approved	○	×	○	○	×	○	○	×
Not_approved	○	×	○	×	×	×	×	○
Not_approved	×	○	○	○	×	×	○	○
Not_approved	×	○	○	×	×	○	×	×

first and the second cubes and the third and the fifth cubes.

Continuity in Attributes which have Ordinal Values

In many knowledge bases, attributes with ordinal values are expressed by a range of values (for example, material which is harder than aluminum should be shaped by grinder-A). Therefore, high continuity of ○s in such attributes leads to high comprehensibility.

Two example knowledge bases for a scholarship approval are shown in Table 3 and Table 4, which have the same cover and the same table size. Values of the attribute *Parent-income* are ordinal values.

These examples implicitly embody the meaning that students whose parent income is low are granted scholarship, which can be seen clearly in Table 3. By contrast, the first cube in Table 4 shows that some students are granted scholarships, if the parent income is less than $60,000 or more than $80,000. This gives the initial impression that anyone with an income of $60,000-80,000 can not be approved. By examining other cubes this can be seen to be false, but this is time-consuming; Table 3 is preferable to Table 4.

Conformity between Concept Functions and Real Cases

In some knowledge bases, there is a dependency relationship between the attributes, which the authors can divide into *precondition* and *constrained* attributes. Whether the constrained attribute relates to concept definition or not depends on the values of the precondition attribute. In such a situation, the positions of the Don't-Cares are critical, because cases that never happen in the real world may be described in knowledge bases.

Other knowledge bases relating to an earned credit at a university are shown in Table 5 and Table 6; they have the same cover and the same table size. These examples implicitly embody the meaning that only students who failed the first exam are eligible to take the makeup exam. There exists an attribute dependency relationship consisting of the precondition attribute *Exam* and the constrained attribute *Makeup-exam*: taking the makeup exam depends on the result of the exam.

In this example, Table 5 is more comprehensible than Table 6, because of the conformity between concept functions and the real cases. In the first cube in Table 6, a case that never happens is described: an examination result is not less than 60 points and the makeup examination result is less than 80 points. Table 5, however, represents only the real cases.

In general, the precondition attributes should not be Don't-Cares, if the constrained attributes are not Don't-Cares.

Table 5: Comprehensible expression for Credit Earning.

	Exam		Makeup-exam	
	≥ 60	<60	≥ 80	<80
Pass	○	×	○	○
Pass	×	○	○	×
Fail	×	○	×	○

Table 6: Incomprehensible expression for Credit Earning.

	Exam		Makeup-exam	
	≥ 60	<60	≥ 80	<80
Pass	○	×	×	○
Pass	○	○	○	×
Fail	×	○	×	○

Algorithm

Figure 1 shows an algorithm to improve the comprehensibility of a decision table. It converts a table with poor comprehensibility to one with high comprehensibility, while preserving logical equivalency.

Table conversion is accomplished by the techniques used in logic minimization algorithm MINI: disjoint sharp operation, Expansion, and Reduction [Hong, Cain, & Ostapko 1974]. In these operations, attributes are required to be ordered, and this order affects concept function forms in a resultant table. The proposed algorithm first determines the attribute order σ, where σ is a list of attributes that specifies the

```
Notation
 a_i:  attribute (1 ≤ i ≤ n)
 C_j:  list of the cubes for jth concept (1 ≤ j ≤ m)
 p:    precondition attribute on attribute dependence relationship
 q:    constrained attribute on attribute dependence relationship
 S:    set of attributes which have ordinal values

 1   begin
             /* Determination of attribute order */
 2       Calculate $n_i$ $(1 \le i \le n)$ by $(U \textcircled{\#} (U \textcircled{\#} C_j))$ with attribute order $(a_i, a_1, a_2 \cdots, a_{n-1})$
 3       $L_{ist}1 \leftarrow$ list of $a_i \in S$ sorted in increasing order of $n_i$
 4       $L_{ist}2 \leftarrow$ list of $a_i \notin S$ sorted in increasing order of $n_i$
 5       $\sigma \leftarrow$ connect $L_{ist}2$ after $L_{ist}1$
 6       for all $(p, q)$ do
 7           if $q$ is placed after $p$ on $\sigma$
 8           then  $\sigma \leftarrow$ list which $q$ is moved to previous position of $p$
 9           endif
10       endfor
             /* Modification of concept functions */
11       for $C_j (1 \le j \le m)$ do
12           $C_j \leftarrow (U \textcircled{\#} (U \textcircled{\#} C_j))$ with $\sigma$
13       endfor
14       Expand and Reduce the cubes with $\sigma$
15   end
```

Figure 1: Algorithm for improving comprehensibility.

order, by some heuristics (in Lines 2-10, Fig. 1), and next modifies the concept function forms by MINI's techniques (in Lines 11-14, Fig. 1).

The main difference between MINI and the proposed algorithm is heuristics to determine the attribute order in each algorithm. While the heuristics in MINI algorithm are mainly for reducing the number of cubes, those in the proposed algorithm are for improving comprehensibility.

For the explanation of heuristics here, consider a decision table constructed solely by min-terms like Table 7. The heuristics for attribute order σ is due to the three out of four comprehensibility criteria:

1. **Table size.**

   A small-size table can be obtained by merging as many min-terms as possible. The algorithm prescans the table and examines the merging ability of each attribute, measured by n_i, where the number of cubes after merging min-terms for all concepts only on attribute a_i. For example, $n_{Bonding-area} = 7$, as shown in Table 8. Also, $n_{Material} = 10$, $n_{Usage} = 8$. Attributes are ordered in increasing order of n_i (in Lines 2-4, Fig. 1). The algorithm merges the cubes, one attribute at a time, in this order. In other words, first the cubes are merged on attribute with high merging ability and last on attribute with low ability. Table 1 is generated by merging the cubes in Table 7 with the σ = (*Bonding-area*, *Usage*, *Material*).

2. **Continuity in attributes which have ordinal values.**

   Attributes with ordinal values are placed at the beginning of σ (in Line 5, Fig. 1). Cubes are first merged on those attributes, generating the maximum number of Os in those attributes. As a result, high continuity of Os in those attributes can be achieved.

3. **Conformity between concept functions and real cases.**

   Attribute order is changed so that the constrained attribute is placed before the precondition attribute (in Lines 6-10, Fig. 1). This change prevents Don't-Cares being generated in the precondition attribute, because cubes are merged on the constrained attribute before considering the precondition attribute. In the example of knowledge bases for credit earning, Table 5 and Table 6 are generated by σ = (*Makeup-exam*, *Exam*) and σ = (*Exam*, *Makeup-exam*), respectively.

After determining the attribute order σ, concept function forms are modified. In the modification, to achieve high Similarity among concept functions, other heuristics are applied.

4. **Similarity among concept functions.**

   High similarity can be achieved, when cubes for many concepts are merged on the same attributes. The algorithm merges the cubes in the same order of attributes for all concepts.

   If, in the early stage of merging, cubes are merged on the attributes, in which the cubes only for the specific concepts can be merged, then similarity becomes low. In Table 7, if first merged on *Material*, cubes mainly for concept Bond-B can be merged. Attribute order σ = (*Material*, *Usage*, *Bonding-*

Area) leads to Table 2, whose similarity is low. However, it is expected that such merging would be prevented, because the cubes are first merged on the attributes which many cubes can be merged (See 1.).

If all cubes in the given table were converted to min-terms for the pre-scan and the modification, the algorithm would take exponential time. To reduce this to a modest amount of computational time, it uses disjoint sharp operation $F \textcircled{\#} G$, where F and G are lists of cubes (details are shown in [Hong, Cain, & Ostapko 1974]).

In the modification, the following feature of the $\textcircled{\#}$ operation is utilized: $U \textcircled{\#} G$ with σ generates more Os in the attribute placed in the earlier position of σ, where U is universe. This operation can generate almost the same concept function forms, as merging min-terms in the attribute order σ. However, since the order of the cubes in C_j affects the number of cubes generated by $U \textcircled{\#} C_j$, the generated table may be redundant. To reduce the table size, cubes are Expanded and Reduced, using the σ order (in Line 14, Fig. 1).

Table 7: Table constructed only by min-terms.

	Material			Usage		Bonding-area	
	Paper	Leather	Plastic	Normal	Industrial	Large	Small
Bond-A	O	×	×	O	×	O	×
Bond-A	O	×	×	O	×	×	O
Bond-A	O	×	×	×	O	O	×
Bond-A	O	×	×	×	O	×	O
Bond-B	O	×	×	O	×	O	×
Bond-B	O	×	×	O	×	×	O
Bond-B	O	×	×	×	O	O	×
Bond-B	O	×	×	×	O	×	O
Bond-B	×	O	×	O	×	O	×
Bond-B	×	O	×	O	×	×	O
Bond-B	×	O	×	×	O	O	×
Bond-C	×	×	O	O	×	O	×
Bond-C	×	×	O	O	×	×	O

Table 8: Table after merging min-terms on attribute *Bonding-area.*

	Material			Usage		Bonding-area	
	Paper	Leather	Plastic	Normal	Industrial	Large	Small
Bond-A	O	×	×	O	×	O	O
Bond-A	O	×	×	×	O	O	O
Bond-B	O	×	×	O	×	O	O
Bond-B	O	×	×	×	O	O	O
Bond-B	×	O	×	O	×	O	O
Bond-B	×	O	×	×	O	O	×
Bond-C	×	×	O	O	×	O	O

Experimental Results

To evaluate concept function forms generated by the proposed algorithm, the authors experimented on 11 real knowledge bases. In addition, they evaluated table size and computational time, using 1 real knowledge base and 24 artificial ones, which are quite large.

Concept Function forms

The proposed algorithm is based on MINI. However, MINI's goal is logic minimization, not knowledge base modification, and the knowledge bases minimized by MINI are incomprehensible.

To confirm the comprehensibility of the generated table, the authors experimented on 11 real knowledge bases, which have 3-7 attributes, 5-20 cubes, 2-6 concepts, and 6-20 columns. Four examples contain attributes with ordinal values and attribute dependency relationships.

Comprehensibility is evaluated by comparing the concept function forms modified by a human with those modified by the algorithm. In seven examples, concept functions produced by the algorithm exactly corresponded to those produced by a human. In the other four examples, results were different, but they were comprehensible to humans.

This difference is mainly due to the limitation in the algorithm: it can only generate cubes which are mutually disjoint. Moreover, the difference might partly be attributed to the heuristics for determining the attribute order. If the calculated n_i values were equal in some attributes, the algorithm would determine the order arbitrarily; it is not guaranteed that expected concept functions are obtained. This situation was observed in two knowledge bases.

Table Size

Experimental results on 24 artificial knowledge bases showed that the algorithm performs logic minimization well. These knowledge bases have 1 concept, 10 cubes, 30 attributes, and 120 columns.

In 21 knowledge bases, size of the generated tables were exactly the same as that by MINI. However, in the other three knowledge bases, MINI was able to generate one or two cubes less than the proposed algorithm. This arises from another limitation in the algorithm: Don't-Cares are collected in the specific attributes, placed in the early position of σ.

The algorithm was also evaluated on a real knowledge base for a grinder selection problem, which has 158 concepts, 1023 cubes, 16 attributes, and 222 columns. In this experiment, the proposed algorithm generated the same number of cubes as MINI.

Computational Time

Pre-scan and modification of a table are based on the disjoint sharp operation. However, it is difficult to estimate the exact cost for the disjoint sharp operation. This is because the cost depends on the concept cover, the attributes order, and the cube order in the right-side cube list of the $\textcircled{\#}$.

To confirm the feasibility of the algorithm, the authors experimented on 24 artificial knowledge bases, described in the previous subsection, with a 33 MIPS work-station. The tables were generated in 210 seconds on an average , which is about 50 % of MINI.

The authors also experimented on a real knowledge base for the grinder selection problem. The resultant table can be obtained in 90 seconds (80 seconds by MINI). This time is not too long and actually much shorter than the time required for modification by a knowledge engineer.

Related Work

Inductive learning algorithms, like ID3 [Quinlan 1986], can also improve the comprehensibility of concept functions. However, the produced concept functions are often incomprehensible, because of the lack of background knowledge and the comprehensibility criteria; they only use the description length criteria implicitly. From another viewpoint, most induction algorithms generate decision trees, not decision tables. Generated decision trees may have a minimum number of nodes and leaves. However, this does not mean a minimum number of cubes.

EG2 [Núñez 1991] uses background knowledge, which is IS-A hierarchy of values, to simplify the decision tree and to obtain a more comprehensible knowledge expression. However, EG2 requires much background knowledge for such simplification. In the proposed algorithm, the only background knowledge required concerns the attribute dependency relationships and the orderings of ordinal attribute values.

The orderings of the ordinal attribute values is used in INDUCE [Michalski, Carbonell, & Michell 1983] to generalize concepts. The proposed algorithm does not generalize the concepts, but produces the logically-equivalent concept functions to those described by knowledge engineers.

Conclusion

This paper presented new comprehensibility criteria regarding concept function forms, and an algorithm for automatically producing comprehensible forms of concept functions. This algorithm is implemented on an expert-system shell DT, which handles classification problems on the decision table [Koseki, Nakakuki, & Tanaka 1991]; its usefulness has been demonstrated in several real problems. Since the concept functions on decision table format can be easily converted to the one on production rule format, this algorithm can be applied to the knowledge bases constructed by production rules.

The new criteria are general ones, which can be applied to many knowledge bases. However, comprehensibility criteria differ according to people and domains, and generated tables may not correspond exactly to the tables expected by knowledge engineers. This disagreement, however, can be overcome by a knowledge editor on DT.

Acknowledgement

The authors express their appreciation to Takeshi Yoshimura of NEC Corporation for giving them the opportunity to pursue this research.

References

[Cragun 1987] Cragun,B.J. 1987. A decision-table-based processor for checking completeness and consistency in rule-based expert systems. *International Journals of Man-Machine Studies* 26(5):3-19.

[Hong, Cain, & Ostapko 1974] Hong,S.J.; Cain,R.G.; and Ostapko,D.L. 1974. MINI:A Heuristic Approach for Logic Minimization. *IBM Journal of Research and Development* :443-458.

[Coulon & Kayser 1978] Coulon,D.; and Kayser,D. 1978. Learning criterion and inductive behavior. *Pattern Recognition* 10(1):19-25.

[Koseki, Nakakuki, & Tanaka 1991] Koseki,Y.; Nakakuki,Y.; and Tanaka,M. 1991. DT:A Classification Problem Solver with Tabular-Knowledge acquisition. *Proceedings of Third International Conference on Tools for Artificial Intelligence* 156-163.

[Michalski, Carbonell, & Michell 1983] Michalski,R.S.; Carbonell,J.G.; and Michell,T.M. 1983. A Theory and Methodology of Inductive Leaning, *Machine Learning: An Artificial Intellegence Approach* Chapter 4, Tioga Press, Palo Alto, 83-134.

[Nguyen et al. 1985] Nguyen,T.A.; Perkins,W.A.; Laffey,T.J.; and Pecora,D. 1985 Checking an Expert Systems Knowledge Base for Consistency and Completeness. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* 375-378.

[Núñez 1991] Núñez,M. 1991. The Use of Background Knowledge in Decision Tree Induction, *Machine Learning* 6(3):231-250.

[Quinlan 1986] Quinlan,J.R. 1986. Induction of Decision Trees. *Machine Learning* 1(1):81-106.